Amendments to the Specification

Please amend the specification of the above-identified application, as follows. In accordance with current amendment practice, changes are shown by underlining (for added matter) and strikethrough (for deleted matter), with the exception that double brackets are used to indicate deleted matter when strikethrough cannot be easily perceived.

Kindly amend the title of the invention as follows:

UNIVERSAL LOAD ADDRESS/VALUE PREDICTION USING STRIDE-BASED PATTERN HISTORY AND LAST-VALUE PREDICTION IN A TWO-LEVEL TABLE SCHEME

Amend paragraph 0002, page 1, as follows:

[0002] The present invention relates to performance improvements in superscalar computer systems. In particular, it relates to an improved method and system for ~~hybride~~ hybrid address prediction.

Amend paragraph 0007, page 2, as follows:

[0007] The simplest algorithm used in prior art value predictors is based on the assumption that the contents of memory locations and registers remains mostly unchanged. So, an appropriate prediction scheme is simply to predict the last value. The so-called last value predictor, further referred to herein as LVP, ~~too, is exemplarily~~ as depicted preferably in Fig. 1, ~~and~~ comprises a table 10 which is addressed by hashing 12 of the instruction address with each entry consisting of a tag field 14 and a last value field 16 ~~for the last value~~.

Amend paragraph 0008, page 3, as follows:

[0008] The table is most likely organized as n-way set associative (e.g. n=4). If a match is found by ~~finding out~~ determining that the tag field matches the instruction address, then the corresponding last value from this table entry is used for prediction. If there is no match, a new entry is made, replacing the Least Recently Used (LRU) table entry as determined by ~~a so-called~~ an LRU algorithm.

Amend paragraph 0009, page 3, as follows:

[0009] ~~Anyway~~Regardless, the predictor is updated each time with the correct value, if it is ~~known~~/confirmed.

Amend paragraph 0011, page 3, as follows:

[0011] Two additional fields, the stride field 20 and a status field 22 are added to each table entry. The idea behind this predictor is that often memory contents are changed by a certain delta value, i.e. a stride. Thus, the next predicted value can be calculated by simply adding the stride to the last value. The status field is used to determine whether the predictor should predict the last value or the last value increased by a certain stride. So the stride predictor further referred to herein as SP is involved only if a certain stride could be found and confirmed ~~whereby~~ as indicated by the status field ~~keeps track of that~~.

Amend paragraph 0014, page 4, as follows:

[0014] It should be noted that such a confirmation is advantageously done when the same stride ~~re-occurs~~ reoccurs at least twice subsequent to each other.

Amend paragraph 0015, page 4, as follows:

[0015] ~~Despite that~~ Although the LVP and SP methods can achieve correct prediction rates of up to more than 50%[[.]], for ~~For~~ certain cases there are still some instructions which alter the contents of memory locations according to a particular pattern which is repeated several times. Therefore, ~~So~~ values can be predicted out of such a context and a ~~so-called~~ context predictor, further referred to herein as CP~~, too,~~ has been proposed as well.

Amend paragraph 0016, page 4, as follows:

[0016] Whereas the SP is an extension of the LVP, the context predictor (CP) is based on a two-table lookup and thus consists of two tables as is illustrated in Fig. 3.

Amend paragraph 0017, pages 4-5, as follows:

[0017] The entries in the first table 30, which is organized as n-way set associative, each comprise a tag field 14, several (e.g. four) last value fields 31a-31d, a LRU ~~info~~ field 32 and a value history pattern field 33. An entry is selected via hashing 12 of ~~e.g.~~ an instruction address. If no match is found, a new entry is added to the table replacing the least recently used table entry according to the LRU field ~~info~~. Specifically, the ~~Said~~ step of adding a new entry ~~in particular~~ comprises: writing ~~to write~~ the tag ~~info~~ information-e.g. the instruction address- in the tag field; writing, ~~to write~~ the current result produced by the instruction in one of the value fields 31a - 31d, and initializing the value history pattern stored in fields 33.

Amend paragraph 0018, page 5, as follows:

[0018] The value history pattern describes the history of the last several (e.g. six) values of the selected memory location used in a series whereby each of the value fields 31a - 31d is identified by a two bit pattern. '00' refers to the value stored in the value field 0, '01' refers to the value stored in the value field 1, etc. For example, if the six most recently used values of a certain instruction were placed in value fields 0,1,2,0,3,2 the corresponding value history pattern (VHP) is '00 01 10 00 11 10'. The LRU field stored in each table entry determines which value field is overwritten ~~replaced~~ if a new value is detected for that instruction.

Amend paragraph 0019, page 5, as follows:

[0019] The two-table lookup is ~~done~~ executed by using the VHP (e.g. a 12-bit pattern) as an address to select an entry in the second table, the ~~so-called~~ pattern history table 34, further referred to herein as PHT, ~~too~~. ~~Exemplarily~~ Preferably, the ~~said~~ second PHT table may have a number of 4K entries in conjunction with the 12-bit pattern used to address this table.

Amend paragraph 0020, page 5, as follows:

[0020] An entry in ~~this~~ the PHT table comprises four saturating 4-bit counters 35a to 35d. These counters represent each value field 31a to 31d in the first table 30. The counter with the highest value and with a count higher than a threshold value selects the appropriate last value stored in the first table. The counters in the PHT are updated according to the current value, i.e.

the corresponding counter is increased by a certain number (e.g., ~~by~~ 3) whereas the other counters are decreased by a certain number (i.e. ~~by~~ 1). The counters saturate (e.g. by 0 resp. 12), and the threshold value (e.g., 6) is chosen ~~(e.g. To be 6)~~ to determine whether a prediction can be made or not.

Amend paragraph 0021, page 6, as follows:

[0021] The second update procedure comprises ~~to update~~ updating the VHP 33[[:]]. Specifically, the ~~The~~ VHP 33 is shifted left two bits and the vacant two bits on the right are filled with the bit pattern corresponding to the current value. If the value was not already stored in one of the 'last value fields', the current value replaces the least recently used last value stored in one of the four value slots and the corresponding two-bit pattern is placed into the VHP 33.

Amend paragraph 0022, page 6, as follows:

[0022] Whereas such a context predictor predicts certain repeating patterns of values -here patterns consisting of up to four different values- it is not effectively predicting strides or last values. Therefore, the best value prediction can be achieved by combining the CP with the LVP/SP. This 'combined' predictor is often called a ~~hybride~~ hybrid predictor (HP). It uses a ~~certain~~ switching scheme to select the predictor of choice in order to achieve the best reliability.

Amend paragraph 0023, pages 6-7, as follows:

[0023] An advantage of the ~~hybride~~ hybrid predictor is that is saves latch counts for using the SP for last value and stride predictions. The major drawback, however, is the complex underlying switching scheme which is necessary in prior art to decide whether to use the LVP or the SP or the CP. According to prior art it is preferred to start the prediction ~~always~~ with the LVP. If the LVP is not successful, but a stride could be found and confirmed, then the SP is invoked. If no stride could be determined, then the CP is initialized and starts collecting and confirming the pattern -assuming that there is a certain pattern of values.

Amend paragraph 0024, page 7, as follows:

[0024] If the pattern stabilizes, i.e., the counters in PHT 33 reach the threshold value, predictions can be made out of context. If the predictor fails, the switching scheme re-enables

the LVP. The disadvantage is, however, that the context predictor invocation is rather inefficient because it takes rather long until the CP is really used because prior to ~~issueing~~ issuing a context prediction the data must be collected which are the basis for a reliable CP.

Amend paragraph 0026, page 7, as follows:

[0026] This objective of the invention is achieved by the features stated in the ~~enclosed~~ independent claims. Further advantageous arrangements and embodiments of the invention are set forth in the respective ~~subclaims~~ dependent claims.

Amend paragraph 0028, page 8, as follows:

[0028] Said new scheme for value prediction ~~allows to predict~~ provides prediction based on last values[[,]] and strides, as well as ~~values out of~~ context prediction, without the use of a sophisticated switching scheme between several predictors. Thus, a quite 'universal' prediction (UP) scheme is disclosed which is based on the two-table lookup mechanism of the context predictor but which deals with differences between subsequent values stored in a certain memory location.

Amend paragraph 0029, page 8, as follows:

[0029] The ~~inventional~~ prediction system of the present invention collects patterns of deltas, i.e., the differences between values, of subsequent values instead of the values ~~itself~~ themselves. Thus, a LVP can be achieved by predicting a 'pattern' of just one stride equal to ~~which is~~ zero. A stride predictor uses a pattern consisting of just one (constant) stride. And a certain pattern of values is ~~now modelled~~ modeled by recording the pattern of deltas between the values and adding the deltas to the last value.

Amend paragraph 0030, page 8, as follows:

[0030] As the context prediction is based ~~now~~ on the deltas, i.e., the differences between some values, the predictor is also capable of predicting values which show a certain pattern of changes. This is thus more general than just recording a certain pattern of values. The main advantage of the ~~inventional~~ context predictor ~~concepts~~ of the present invention is that it inherently involves

the switching scheme ~~inherently~~, i.e., if a certain counter reaches a hit-threshold value, the prediction out of context, including stride prediction, as well as last value prediction is started.

Amend paragraph 0031, page 8, as follows:

[0031] According to a preferred embodiment thereof the default and initial prediction method is LVP by using a stride [[= 0]] equal to zero. This can be achieved by initializing the corresponding counter to the threshold value. If the value is not predictable at all, this counter will be decreased below the threshold and the new status 'not predictable' will be recognized and can be issued. This is a remarkable advantage compared to prior art because the performance penalty due to a misprediction recovery can be remarkably higher than waiting until the dependency is resolved and the result is calculated in an ordinary manner.

Amend paragraph 0033, page 8, as follows:

[0033] The predictor thus saves ~~thus~~ array counts, because the strides stored in the stride fields may have a restricted number of bits compared to the last value stored in the CP. This is true despite the fact that the last value must be stored in an additional field in each entry. Assuming that the values to predict are 64 bits wide and that a stride field consisting of 16 bits is sufficient, four stride fields and the last value field together will consume 128 bits, whereas the CP with four last values stored in each entry will consume as much as 256 bits.

Amend paragraph 0034, page 8, as follows:

[0034] Advantageously, the number of stride fields is greater than 3 and smaller than 7 for ~~being applied~~ application in today's modern computer architectures.

Amend paragraph 0039, page 10, as follows:

[0039] Fig. 4 is a schematic block diagram showing the essential components used in a ~~hybride~~ hybrid predictor according to a preferred embodiment of the present invention~~, and~~

Amend paragraph 0042, page 11, as follows:

[0042] With general reference to the figures and with special reference now to Fig. 4, the essential components used in a ~~hybride~~ hybrid predictor according to a preferred embodiment of the present invention, which is referred to herein ~~under~~ below as 'universal predictor' (UP), are described in more detail ~~next~~ below ~~and~~, by way of example, ~~applying~~ for the prediction of instruction ~~address prediction for~~ addresses having 64 bits ~~bit addresses~~.

Amend paragraph 0043, page 11, as follows:

[0043] The ~~universal predictor~~ UP is a two-level predictor comprising two tables 40 and 44. The entries in the first table 40, which is organized as 4-way set ~~fourfold~~ associative, comprise: a (prior art) tag field 14, (32 bit long); [[,]] a LRU ~~info~~ field 32, 6 bit long, ~~dependent of~~ depending on the number of stride fields in use, a last value field 42, 64 bit long, four stride fields 41a to 41d, each 16 bit long, and a stride history pattern (SHP) field 43, (6 ~~time~~ times 2 bits = 12 bits long).

Amend paragraph 0044, page 12, as follows:

[0044] An entry of ~~said~~ table 40 is selected via hashing 12 of the instruction address. If no match is found, a new entry is added to ~~the~~ table 40 replacing the least recently used entry ~~table~~ according to the ~~another~~ LRU field ~~info~~. ~~This~~ There is a 6-bit pattern for each hashing address which keeps track of the ~~least recently used~~ LRU table entry in ~~of the fourfold set associative organized first~~ table 40.

Amend paragraph 0045, page 12, as follows:

[0045] When a new instruction occurs the first time during an operation, no stride will be known for it, and a new entry must be added. This step ~~which~~ comprises: ~~to write~~ writing the tag info, i.e., the instruction address, into the tag field 14~~, to write~~; writing ~~to write~~ the current value in the last value field 42; writing ~~and to write~~ stride = 0 into one, e.g., the first , of the four stride fields 41a,..41d[[,]] ; and initializing the stride history pattern, by e.g. '00 00 00 00 00 00', if stride = 0 is written into the first stride field. Thus, ~~for~~ the next time, at most a stride=0, i.e., the last value can be predicted. When a stride not equal to 0 turns out to be true, than some delta

exists, and LVP turns out not to be adequate. This ~~and said~~ delta can be taken as the stride for future prediction by replacing the former stride=0 in the ~~stride 0~~ str0 field 41a.

Amend paragraph 0046, page 13, as follows:

[0046] The stride history pattern describes the history of the last six strides used in series where each stride is identified by a two bit pattern, e.g., '00' for the stride placed in the stride ~~filed~~ field 0, '01' for the stride placed in stride field 1, and so on. When for example the six recently used strides were placed in stride fields 0,1,1,0,3,2 then the stride history pattern (SHP) would be 00 01 01 00 11 10.

Amend paragraph 0047, page 13, as follows:

[0047] A second LRU ~~info~~ value stored in the LRU ~~info~~ field 32 of each table entry determines which stride in the stride fields has to be replaced if more than 4 strides are needed and the least recently used stride is replaced.

Amend paragraph 0048, page 13, as follows:

[0048] The ~~two table~~ two-table lookup is then ~~done~~ executed using the stride history pattern SHP (a 12-bit pattern) as an address to select an entry in a second, so-called pattern history table 44 (PHT) having 4 K entries. An entry in this table comprises four saturating 4-bit counters. Each counter 45a..45d is associated to a respective stride field 41a..41d in the first table 40. The counter with the highest value and with a count higher than a particular predetermined threshold value selects the ~~appropiate~~ appropriate stride which is used for the prediction. This step ~~which~~ is then ~~done~~ executed like in the prior art -see the bottom portion of Fig. 3 and 4, but is[[-]] based uniformly on strides instead of separately evaluating values, strides and value based patterns. The predicted value is calculated by an addition of the selected stride and the last value. If the counter(s) in the PHT 44 are below said ~~treshhold~~ threshold value, then no prediction will be made, and thus a status 'not predictable' is granted in the respective cycle.

Amend paragraph 0050, page 13, as follows:

[0050] In order to provide a short setup time ~~of~~ for the predictor, the number of requests to a certain table entry ~~until~~ before a prediction for the corresponding instruction ~~can be~~ is made should be ~~held~~ as small as possible. Thus, a particular initialization of the predictor is required.

Amend paragraph 0052, page 14, as follows:

[0052] If the last value is wrong, the current difference is stored as a stride, and the next ~~time~~ prediction can be made using this stride. Despite that, the predictor will still predict the last value until the stride is confirmed.

Amend paragraph 0053, page 14, as follows:

[0053] Without a special initialization, the predictor according to the invention will start to ~~inventional concept proposes to~~ predict only if at least one counter in the PHT ~~will exceed~~ exceeds a certain threshold value. This means that depending on the counter update procedure - comprising in turn increasing ~~of~~ the correct PHT counter and decreasing the remaining counters - ~~it needs~~ several requests to the predictor are needed before ~~until~~ the predictor actually starts the value prediction.

Amend paragraph 0054, page 14, as follows:

[0054] In particular, when a new instruction is found and a new entry is written into the first table the current value is placed in the last value field, stride 0 into the str0 field and the LRU is initialized so that the next stride is written into str0 replacing stride 0. The SHP is initialized with the pattern '00 00 00 00 00 00' which describes a valid history for a last value/stride predictor which always uses the stride stored in str0. This pattern is unique for a LVP/SP and the corresponding counters in the PHT must be set ~~appropriate~~ appropriately to ensure that the prediction will use str0, i.e. the first counter is set to a value well above the threshold (e.g. to the maximum value 12) and the other counters to a value well below the threshold (e.g. = 0).

Amend paragraph 0055, page 14, as follows:

[0055] Assuming that every stride field (str0, str1, str2 or str3) can be used in LVP/SP prediction, the corresponding SHP ("00 00 00 00 00 00", "01 01 01 01 01 01", "10 10 10 10 10 10" or '11 11 11 11 11 11") address certain counters in the PHT which can be initialized (and even fixed) appropriately. ~~appropriate:~~ If the stride used for prediction is stored in stride field str2, the second counter of entry '101010101010' in the PHT is preset to a value well above the threshold and the remaining ~~remaing~~ counters to values well below the threshold value. Accordingly, the following PHT entries can be preset (and even fixed) to the following counter values:

SHP = PHT-address	PHT-cnt0	PHT-cnt1	PHT-cnt2	PHT-cnt3
00 00 00 00 00 00	12	0	0	0
01 01 01 01 01 01	0	12	0	0
10 10 10 10 10 10	0	0	12	0
11 11 11 11 11 11	0	0	0	12

Amend paragraph 0059, page 17, as follows:

[0059] In this way, ~~an immediate response of~~ the ~~inventional~~ prediction method of the instant invention provides an immediate response to the neutral ~~start~~ starting conditions, as well as to the initial values ~~creation~~ of new table entries ~~can be achieved~~.

Amend paragraph 0061, page 17, as follows:

[0061] In a first step 510 - when the program is started- all counters are initiated, i.e. setup, [[-]] ~~advantageously~~ according to the scheme given above.

Amend paragraph 0062, page 17, as follows:

[0062] When a result is available from a newly completed instruction, see ~~yes branch~~ yes-branch of decision 520, it is checked, see decision 530, to determine if the same instruction can be identified as ~~to be~~ present in table 40. Thus, the tag field 14 in table 40 is checked and the tag compared with the instruction address. As long as no result is available, [-] see the ~~no branch~~ no-branch of decision 520, [[-]] control is fed back to repeat the check of decision 520.

Amend paragraph 0063, page 18, as follows:

[0063] In the no-branch of decision 530, i.e., when no matching entry ~~was~~ is found, said current instruction is installed in the first table 40, see block 540. [[:]] In particular, the tag field 14 is written, the SHP field 43 is setup, ~~as well as~~ the LRU field 32 is initialized, and a stride of 0 is written into stride field 41a of the respective new entry in table 40.

Amend paragraph 0064, page 18, as follows:

[0064] ~~Else~~ Otherwise, in the yes-branch ~~yes-branch~~ of decision 530, the current stride is calculated by subtracting the last value from the current result, see step 550.

Amend paragraph 0065, page 18, as follows:

[0065] Then, at decision 555, it is ~~checked~~ determined if the ~~same~~ current stride can be found in one of the stride fields 41a,.. 41d~~, decision 555~~.

Amend paragraph 0066, page 18, as follows:

[0066] If not, ~~in~~ the no-branch ~~no branch~~ of decision 555 is followed and [[-]] the current stride is stored into the respective stride field which is specified by the value store in the LRU field 32 and said LRU field is updated[[,]]; see block 560.

Amend paragraph 0067, page 18, as follows:

[0067] In the yes-branch of 555 a current stride was already stored in one of the stride fields. Now, as well as after performing block 560, the corresponding PHT counters 45a,.. 45d are updated, in block 565, by increasing the correct counter by 3 and decrementing the other counters by 1. It should be noted that the respective entry in table 44 is addressed by the SHP in field 43.

Amend paragraph 0068, pages 18-19, as follows:

[0068] Then, as well as after performing block 540, the new stride history pattern is calculated as described further above, see step 570. In particular, the SHP field 43 is shifted left by two bits and the vacant bits on the right are replaced by the bit pattern corresponding to the current

correct stride. If this stride is not found, the current stride is written ~~for replacing~~ to replace the least recently used stride field, and the corresponding 2-bit pattern is placed in the SHP 43.

Amend paragraph 0069, page 19, as follows:

[0069] Finally, the result is stored in the last value field 42, see step 575, and control is fed back to decision 520 in order to process the next instruction upon its completion.

Amend paragraph 0070, page 19, as follows:

[0070] With reference now to Fig. 6 the prediction procedure is described in more detail. It should be noted that - in ~~said~~ the preferred embodiment - the update/setup procedures and the now described prediction procedure are implemented as independently running processes which access the same hardware arrangement by respective write (Fig. 5) and read accesses (Fig. 6), respectively.

Amend paragraph 0072, page 19, as follows:

[0072] In [[a]] step 610, the instruction is first decoded. Then, in decision 620, it is determined ~~checked in a decision 620,~~ if the same instruction can be identified to be present in table 40. Thus, the instruction address is compared with the tag stored in tag field 14 in table 40.

Amend paragraph 0073, pages 19-20, as follows:

[0073] If no matching instruction is found, no prediction is possible[[,]] (see block 630), and the status 'not predictable' ~~can advantageously be~~ is signaled ~~signalled, i.e., issued in order~~ to prevent [[a]] an error in prediction ~~misprediction~~, see step 635. Then the control is fed back to step 610, again, for decoding the next instruction~~, to step 610, again~~.

Amend paragraph 0074, page 20, as follows:

[0074] Otherwise, if a matching instruction is found (i.e. there is a tag hit), the yes-branch of decision 620 is followed such that~~, after tag hit,~~ the stride history pattern is read from field 43 of the first table 40, ~~field 43,~~ see step 640. This pattern is used for selecting a respective matching entry in the second table 44 in order to evaluate and select the counter values, see step 650.

Amend paragraph 0075, page 20, as follows:

[0075] Thus, the counters and the corresponding patterns can be read and evaluated, in particular, to determine if any counter's current count is above a predetermined threshold value~~,~~ of , for example ~~e.g.~~ 6, see decision 670.

Amend paragraph 0076, page 20, as follows:

[0076] If, in the yes-branch of 670, a counter has a count of greater than the threshold value of, for example, six (6) [[6]] the respective prediction can automatically be undertaken by selecting the highest counter, see step ~~660~~ 680. This is a remarkable advantage compared to prior art which needs a complicated switching scheme in order to change from LVP to SP, and in particular from SP to CP.

Amend paragraph 0078, page 21, as follows:

[0078] In the foregoing description the invention has been described with reference to a specific ~~exemplary~~ preferred embodiment thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. Accordingly, the ~~The~~ specification and drawings are ~~accordingly~~ to be regarded as illustrative rather than ~~in a~~ restrictive ~~sense~~.

Amend paragraph 0079, page 21, as follows:

[0079] In particular, the dimensions of the fields given in the above ~~exemplary~~ preferred embodiment may be varied as required, depending on the computer ~~for the respective~~ processor architecture in use.